Alexandre DUFOUR (1054564)

Camille CLAING (20022341)

Nahuel LONDONO (20097185)

Raphaël LAJOIE (20099664)

IFT 2935 - Bases de données

Projet

Gestion d’un système de vente conditionnelle

Travail présenté à

M. Michel Boyer

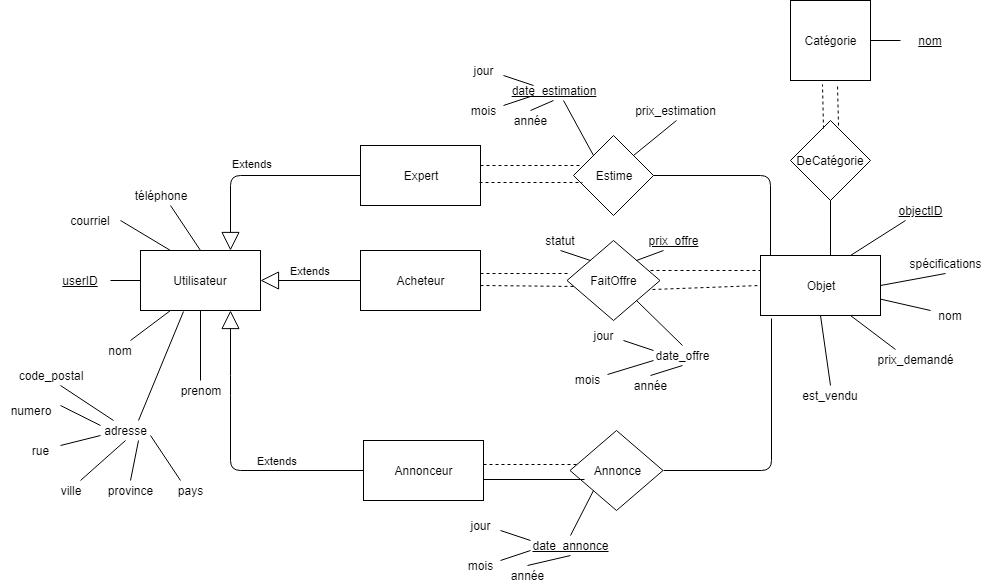
Département des arts et des sciences

Discipline informatique

Université de Montréal

10 avril 2019

1. **Modèle Entité-Associations**



**Explications :**

- On considère que chaque produit ne peut être estimé qu’une seule fois et qu’un produit ne peut être estimé que par un seul expert. C’est pourquoi le prix de l’estimation est un attribut du produit, et non de l’estimation.

- On considère que l’achat d’un produit est fait par le premier utilisateur ayant fait une offre égale ou supérieur au prix de l’estimation. Si deux offres acceptables sont faites en même temps, mais que le prix est différent, la vente est faite à l’utilisateur ayant fait l’offre la plus haute.

- Les catégories possibles sont {Meubles, Multimédia, Informatique, Article de bureau, Autres}. Puisqu’on veut garantir que l’utilisateur ne puisse pas entrer une catégorie inexistante pour un produit, on a créé une entité *Catégorie* qui énumère les catégories de produit possible.

- On considère qu’un produit ne peut faire partie que d’une seule catégorie.

- La variable « statut » d’une offre peut avoir comme valeur {acceptée, refusée, en attente}. Cela permet de clarifier le statut d’une offre faite par un acheteur.

1. **Modèle relationnel**

Utilisateurs (userID, nom, prénom, courriel, téléphone, numeroAdr, rue, ville, province, pays, codePostal)

Objets (objectID, nomObjet, #nomCatégorie, prixDemandé, estVendu, spécifications)

Annonces (#objectID, #userID, jour, mois, année)

FaitOffre (#objectID, #userID, prixOffre, statut, jour, mois, année)

Estimations (#objectID, #userID, jour, mois, année, prixEstimation)

Catégories (nomCatégorie)   
  
----------------------------------------------------------------------------------------------------------------------------------------------------------------

1. **Requêtes SQL (voir LMD.sql)**

**3.1) Donne la liste des objets vendu par les utilisateurs.**

WITH objVendu AS (SELECT userID, nomObjet FROM Produits WHERE (estVendu = 'true')

SELECT \* FROM Utilisateurs NATURAL JOIN objVendu;

**3.2) Donne les utilisateurs n’ayant pas d’objet à vendre.**

SELECT nom, prenom, userID FROM Utilisateurs WHERE NOT EXISTS

(SELECT \* FROM Annonces WHERE Utilisateurs.userID = Annonces.userID);

**3.3) Donne l’utilisateur ayant le plus d’objet vendu.**

WITH nbrVente AS (SELECT userID, count(object\_ID) AS ventes FROM Annonces GROUP BY userID),

maxVente as (SELECT \* from nbrVente where ventes = (Select Max(ventes) from nbrVente))

SELECT nom, prenom, MAX(ventes) FROM nbrVente NATURAL JOIN Utilisateurs;

**3.4) Donne le nombre d’offres faite sur les produits de la catégorie « Multimédia» d’un utilisateur.**

WITH objInfo AS (SELECT object\_ID FROM object WHERE nomCatégorie = 'Multimédia'),

offresInfo AS(SELECT object\_ID, userID, FROM FaitOffre NATURAL JOIN objInfo),

nbrOffre AS(SELECT userID, count(object\_ID) as nombre FROM offresInfo GROUP BY userID)

SELECT nom, prenom FROM Utilisateurs NATURAL JOIN nbrOffre;

**3.5) Donne le ou les utilisateurs ayant fait des offres inférieures au prix de l’estimation pour un produit**

WITH OffresProduit AS (SELECT object\_ID, userID, prixoffre, prixEstimation FROM faitOffre NATURAL JOIN estimation),

OffresValides AS (SELECT \* FROM OffresProduit WHERE prixOffre < prixEstimation),

offresObj as (Select \* from offresValides NATURAL JOIN object),

offresUtil as (Select \* from utilisateurs NATURAL JOIN offresObj)

SELECT nom, prenom, userID, nomObjet, prixOffre, prixEstimation FROM OffresUtil;

**3.6) Donne le nom de l’utilisateur habitant au Canada qui a acheté le plus de produits de la catégorie « Autres ».**

WITH userCanada AS (SELECT userID, nom, prenom FROM Utilisateurs WHERE pays = 'Canada'),

objInfo AS (SELECT object\_ID FROM object WHERE nomCatégorie = 'Autres' AND estVendu = 'true'),

vendu AS (SELECT object\_ID, userID FROM faitOffre WHERE statut = 'acceptée'),

achat AS (SELECT userID, object\_ID, nom, prenom FROM userCanada NATURAL JOIN vendu),

nbrAchat AS (SELECT nom, prenom, count(object\_ID) AS nbAchats FROM achat GROUP BY userID, nom, prenom),

max AS (SELECT Max(nbAchats) AS max FROM nbrAchat)

SELECT nom, prenom, max FROM (nbrAchat INNER JOIN max ON nbrAchat.nbAchats = max.max);

**3.7) Donne tous les identifiants et la catégorie d’objets vendus à plus de 50$, estimés par un expert ayant comme nom de famille «Tremblay ». De plus, les objets doivent être bleus.**

WITH Tremblay AS (SELECT userID FROM utilisateurs WHERE nom = 'Tremblay'),

experts As (SELECT userID, object\_ID FROM Tremblay NATURAL JOIN Estimation),

obj AS (SELECT object\_ID, nomObjet, nomcatégorie, spécifications FROM object WHERE (estVendu = 'true' AND prixdemandé > 50.00 AND spécifications LIKE '%bleu%'))

SELECT userID, nomcatégorie, nomObjet, spécifications FROM experts NATURAL JOIN obj;

**3.8) Donnez toutes les informations relatives à l’adresse des acheteurs qui ont fait une annonce, au Boxing day (le 26 décembre 2017 ou 2018).**

WITH annonces AS (SELECT DISTINCT userID FROM Annonces WHERE ( \_date = '2018/12/26' OR \_date = '2017/12/26')),

bonAcheteurs AS (SELECT codepostal, numeroAdr, rue, ville, province, pays, userID FROM Utilisateurs NATURAL JOIN annonces)

SELECT codepostal, numeroAdr, rue, ville, province, pays FROM bonAcheteurs;

**3.9) Pour la catégorie « Article de bureau », donner la liste des objets qui ont n’ont pas été achetés en ordre croissant de prix. De plus, l’annonceur doit habiter au Québec.**

WITH objBureau AS (SELECT object\_ID, nomObjet, prixdemandé, spécifications FROM object WHERE (nomCatégorie = 'Article de bureau' AND estVendu = 'false')),

objet\_seller AS (SELECT nomObjet, spécifications, userID, prixdemandé FROM annonces NATURAL JOIN objBureau),

Queb AS (SELECT userID FROM utilisateurs WHERE province = 'QC')

SELECT nomObjet, spécifications, prixdemandé FROM objet\_seller NATURAL JOIN Queb Order By(prixdemandé);

**3.10) Donner la liste des offres sur les lits noirs (qui ont été achetés).**

WITH produitsPrécis AS (SELECT object\_ID FROM object WHERE (estVendu = 'true' AND spécifications LIKE '%Lit%noir%'))

SELECT \* FROM produitsPrécis NATURAL JOIN FaitOffre;

1. **Tables et exemplaires (voir LDD.sql)**

4.1 Justifications des tables

* On a décidé de regrouper tous les utilisateurs dans une même table, car certains utilisateurs peuvent avoir plusieurs rôles. Par exemple, un annonceur peut décider, après son annonce, d’être acheteur.
* La table des objets sert à regrouper tous les objets, car les éléments mis en vente ont tous des attributs les regroupant.
* On veut garantir que l’utilisateur ne puisse pas entrer une catégorie inexistante pour un produit, on a créé une entité Catégorie qui énumère les catégories de produit possible.
* On a voulu une table pour représenter chaque association, car celles-ci ont des attributs propres à elles. De plus, ça rend les informations sur les offres, estimations et achats plus faciles à accéder.

4.2 Justifications sur le choix des clés primaires

* Les identifiants d’objets et d’utilisateurs sont des attributs très importants dans la sémantique de ce travail. En effet, cela permet de regrouper et classer plus facilement ces éléments lorsqu’on veut y accéder. C’est pourquoi ces identifiants sont considérés comme des clés primaires de nos tables.
* Pour la relation « FaitOffre », on a choisi d’utiliser également l’attribut « prixOffre » comme clé primaire, car on a jugé qu’il y a un lien fort sémantiquement entre l’offre et son prix.

4.3 Autres justifications

* Dans le fichier LDD.sql, on a décidé d’utiliser le format « DATE », prévu dans ce type de fichier, au lieu de séparer le jour/mois/année. C’est tout simplement plus facile à gérer au moment de la lecture de ce fichier.
* De plus, certaines explications sont dans la section 1 (schéma EA) de ce document.

----------------------------------------------------------------------------------------------------------------------------------------------------------------

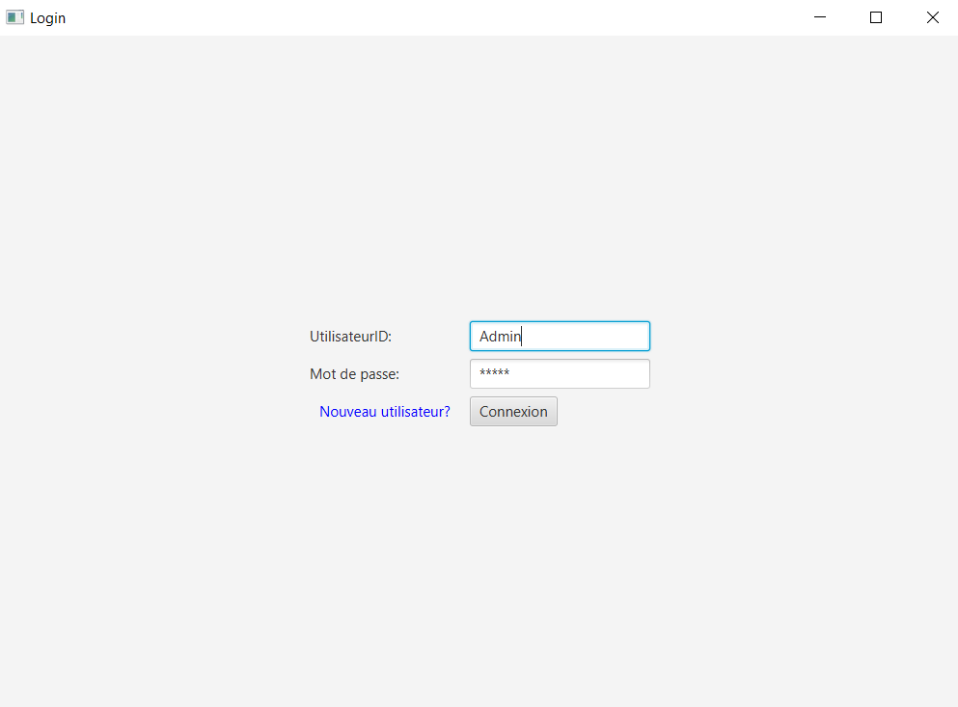
1. **Guide de l’utilisateur**

Pour exécuter le programme, l’utilisateur doit taper en ligne de commande :

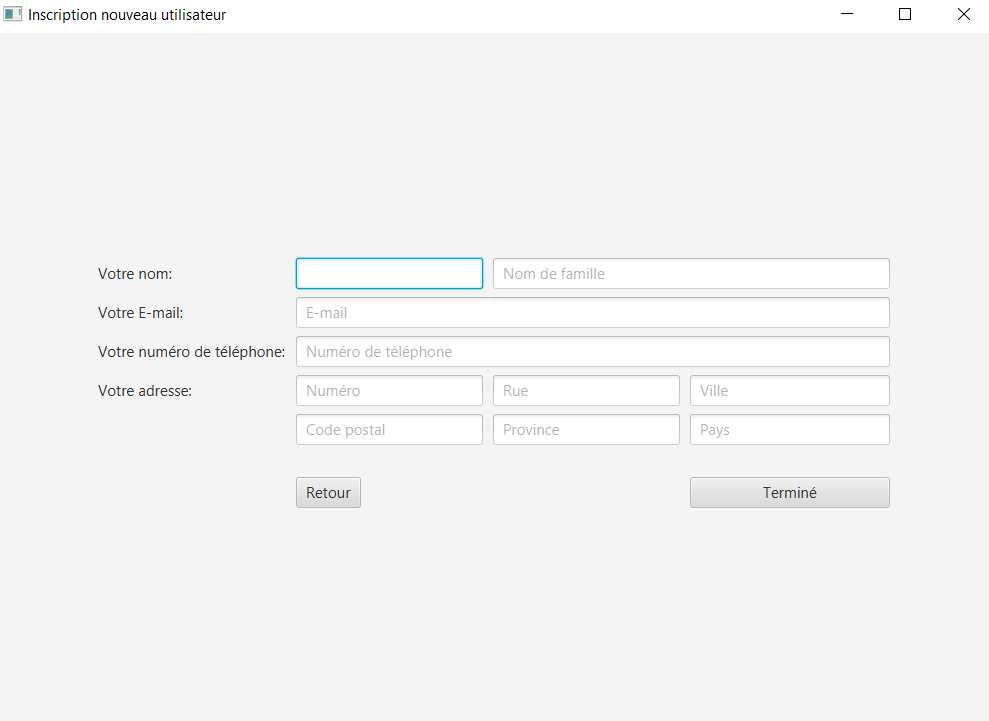
java -jar IFT2935\_Projet.jar

5.1 Les premiers pas dans le programme

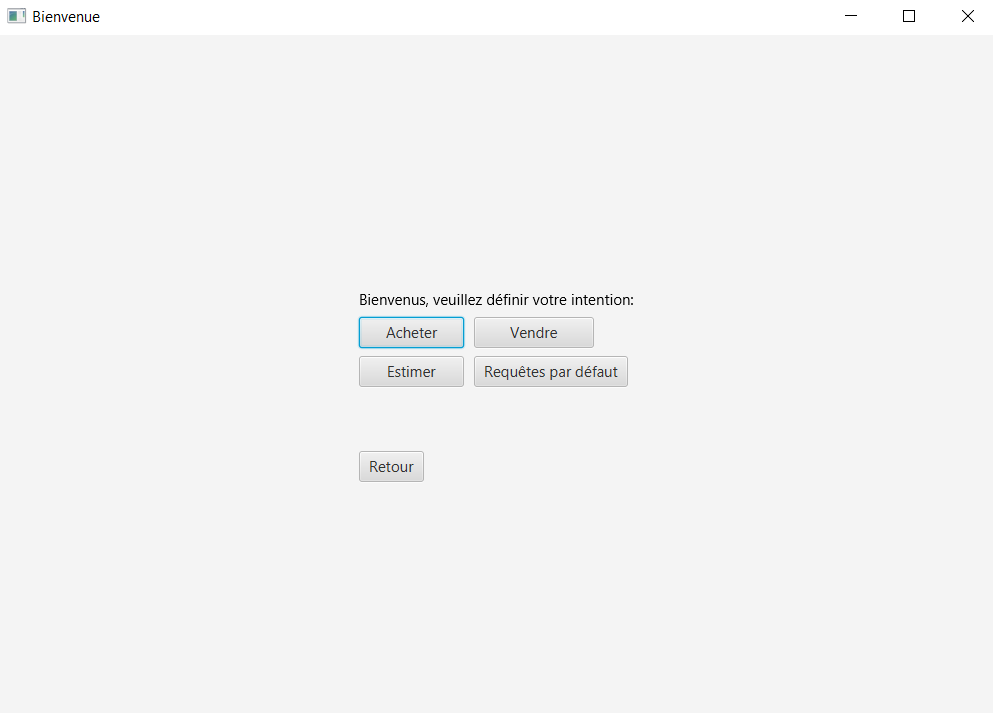
En démarrant le programme, l’utilisateur visualise cet écran. En utilisant l’utilisateurID « Admin » et le mot de passe déjà entré dans la boîte de texte, l’utilisateur pourra avoir accès à tous les droits, c’est-à-dire agir comme acheteur, vendeur, expert ou encore voir des exemples de requêtes de base :



S’il s’agit d’un nouvel utilisateur, celui-ci pourra se créer un nouveau compte. Il clique alors sur « Nouveau utilisateur? » et entre ses informations personnelles. Lui sera assigné un utilisateurID. Il pourra ensuite avoir accès directement à la base de données en cliquant sur « Terminé » :

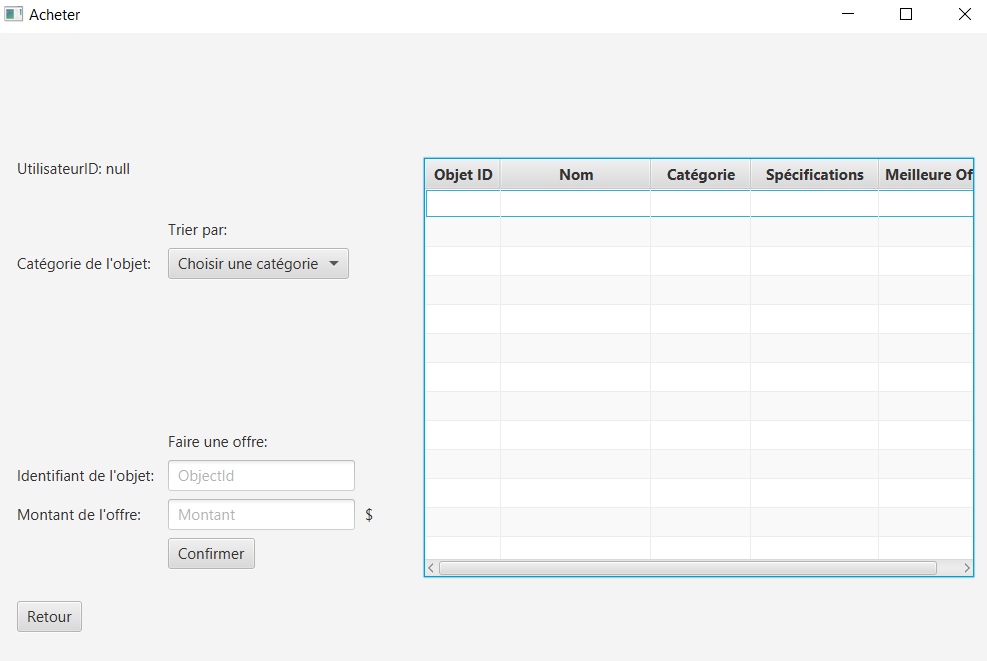


L’utilisateur doit à ce point décider du rôle qu’il désire prendre. Il peut soit acheter un produit, soit afficher un produit à vendre, soit estimer un produit affiché par un vendeur, ou encore visualiser les réponses à 10 des questions les plus fréquentes :



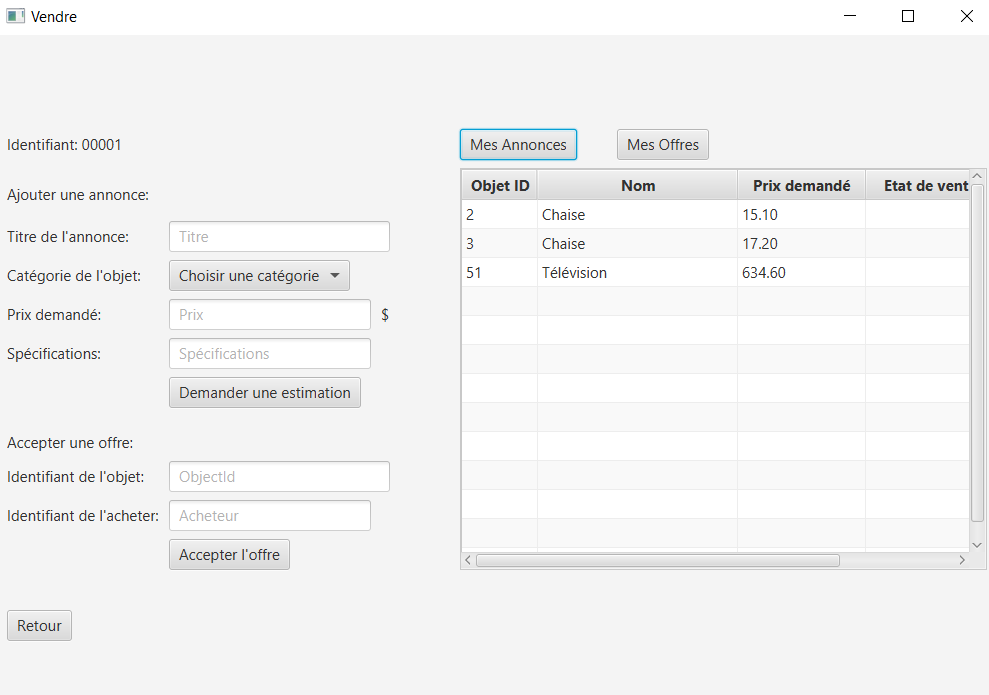
5.2 L’acheteur

Ci-dessous le point de vu de l’acheteur. Celui-ci pourra voir les annonces disponibles à la vente en choisissant une catégorie. S’il trouve un objet pour lequel il désire faire une offre, l’acheteur doit entrer l’identifiant de l’objet et le montant de l’offre qu’il propose pour l’objet. Le vendeur pourra ainsi voir l’offre dans son compte. À noter que si l’offre est supérieure à l’estimation, l’objet lui sera automatiquement vendu.

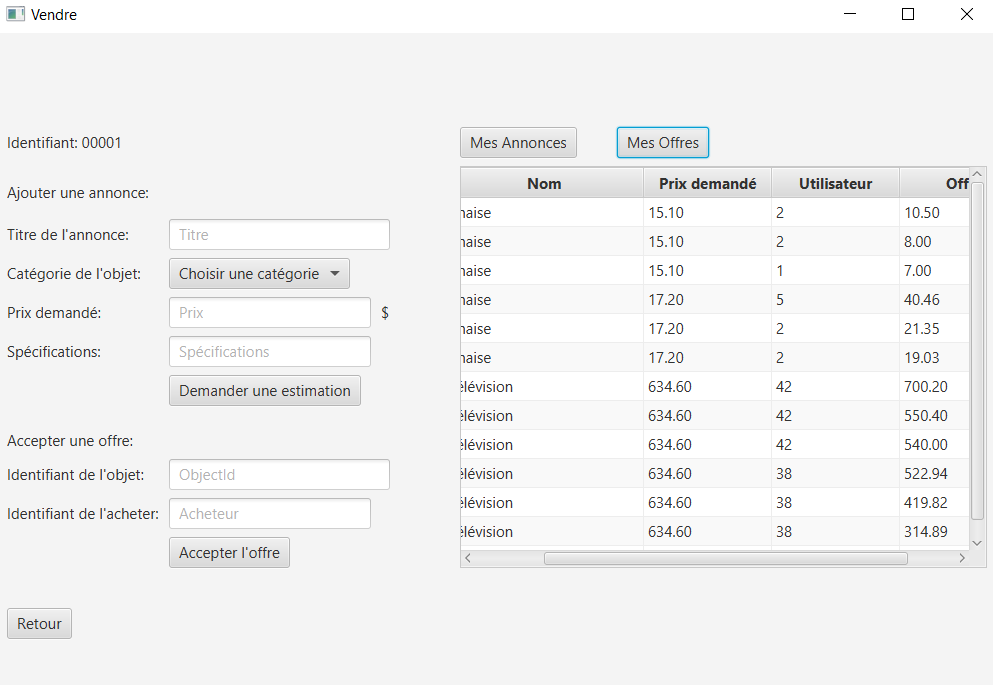


5.3 Le vendeur

Si on retourne à l’écran de choix et qu’on clique sur « Vendre », on tombe avec la vue présente ci-dessous. Il pourra alors créer une nouvelle annonce en complétant les champs nécessaires dans la section « Ajouter une annonce ». En cliquant sur « Demander une estimation », un expert fera une estimation de son objet. Il pourra alors l’accepter ou la refuser. S’il l’accepte, son annonce sera affichée dans la table des annonces et son objet dans la table des objets, visible aux acheteurs. Sinon, un autre expert estimera son objet, et une nouvelle notification lui sera envoyé ultérieurement. De plus, en cliquant sur « Mes Annonces », l’utilisateur voit les informations pertinentes des annonces qu’il a faites :

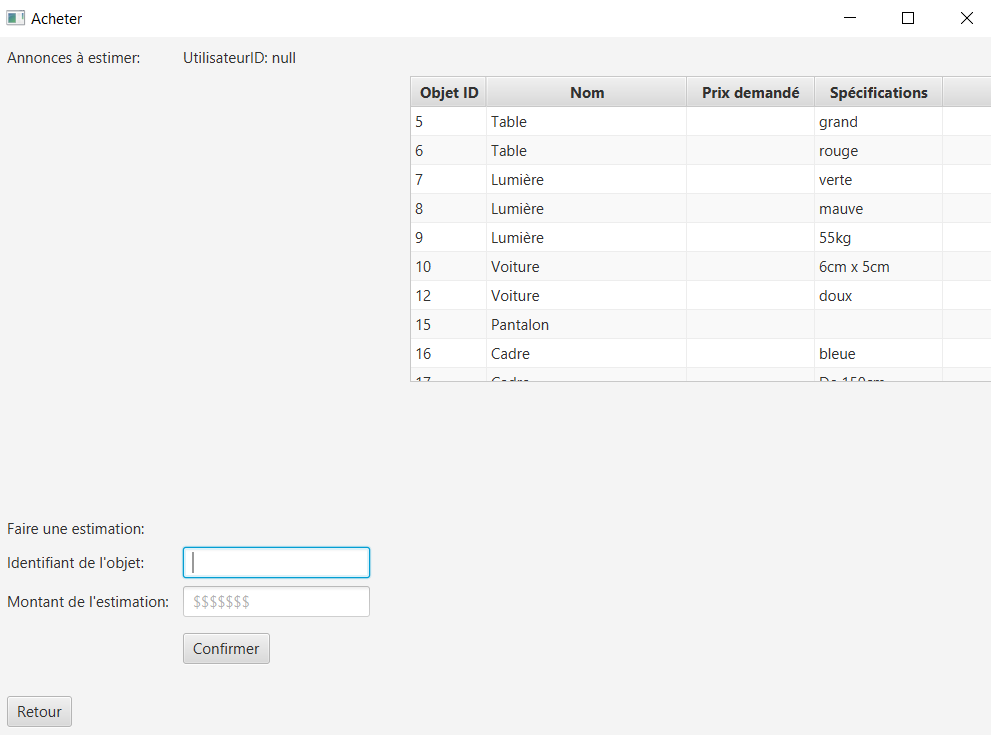


L’utilisateur peut également cliquer sur « Mes offres » pour voir les offres que d’autres utilisateurs ont fait de ses objets. Il pourra alors, s’il le désire, accepter une offre en entrant l’identifiant de l’objet et de l’utilisateur ayant fait l’offre dans la section réservée à cet effet. À noter que si un acheteur fait une offre supérieure au prix de l’estimation pour un objet, ce dernier lui sera automatiquement vendu, et il n’apparaitra pas dans les offres :



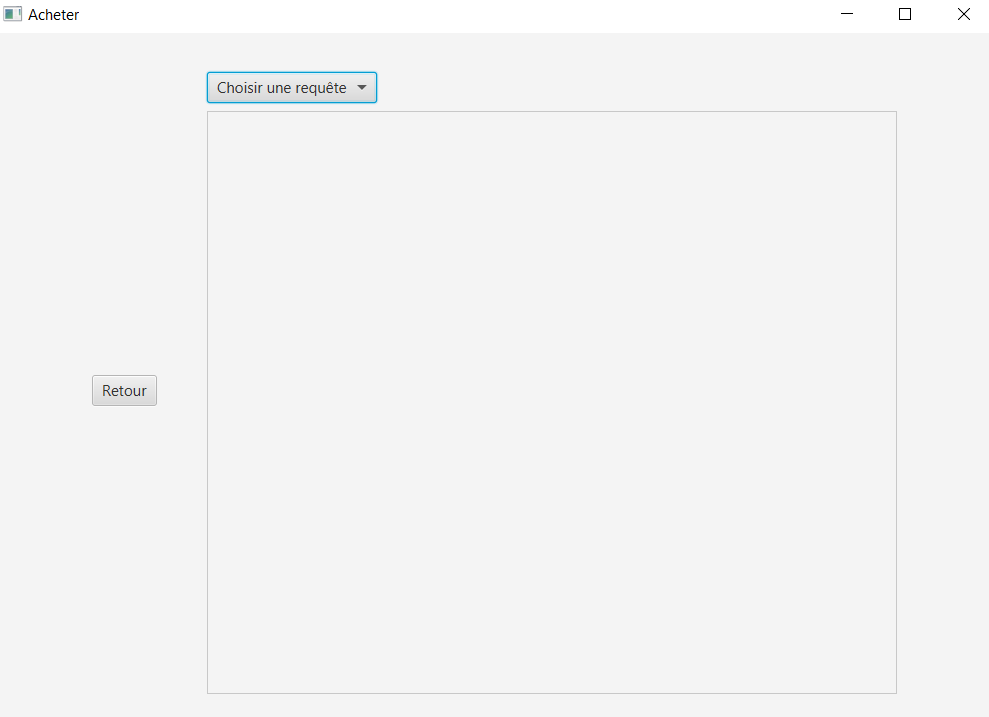
5.4 L’expert

Si on revient à l’écran de choix du rôle de l’utilisateur, et que l’on clique sur estimer, l’utilisateur devient un expert. Il pourra ainsi estimer des objets qu’un autre utilisateur veut mettre en vente. En ouvrant la fenêtre, dans le tableau se trouvent toutes les annonces n’ayant pas été estimée. Selon le titre de l’objet et les spécifications, l’expert pourra déterminer un prix d’estimation. Pour éviter que son choix soit influencé par le vendeur qui a postulé l’annonce, la colonne des prix demandés par le vendeur est masquée. Pour produire une estimation, l’expert ne fait qu’entrer l’objet ID de l’objet, et le montant de l’estimation. Un message est alors envoyé au vendeur, et celui-ci devra décider s’il accepte ou refuse l’estimation :

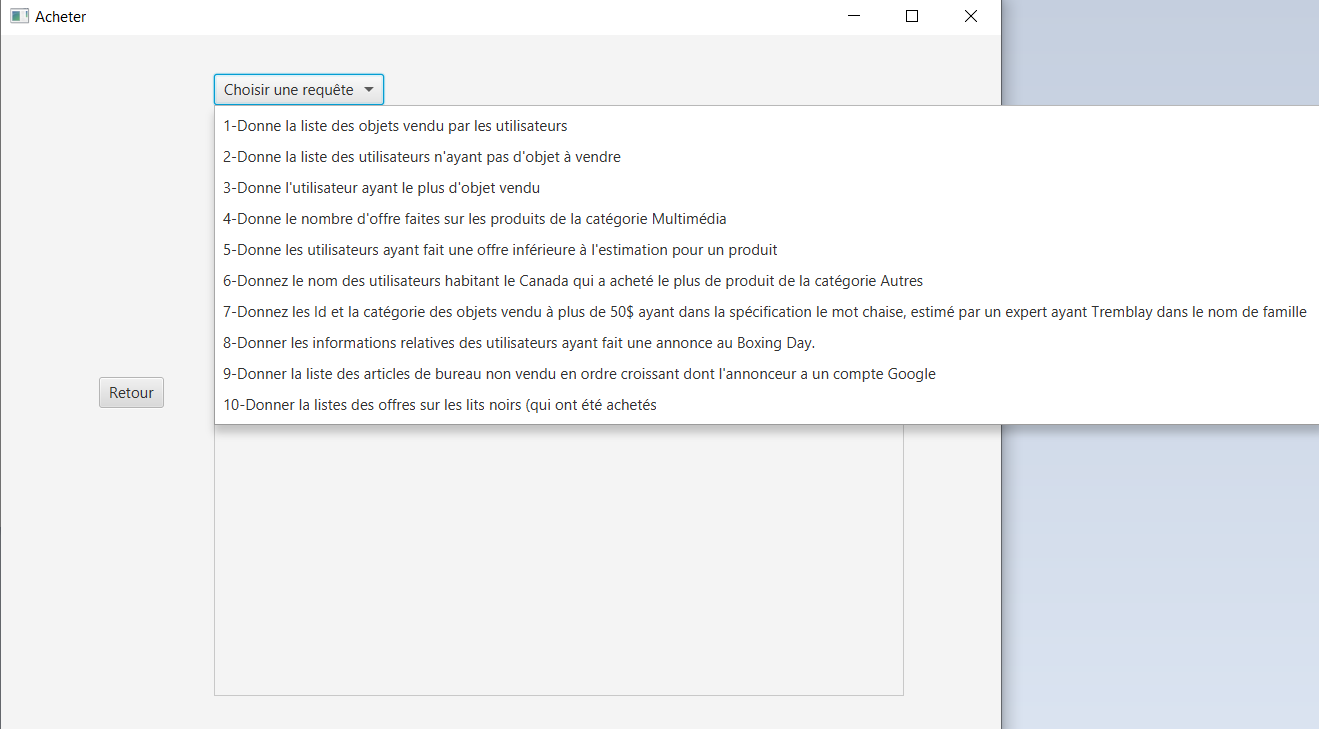


5.5 Les requêtes fréquentes

En cliquant sur « Requêtes par défauts », l’utilisateur observe cet écran. Il lui sera possible ici de voir les résultats de questions fréquemment posées :



L’utilisateur aura ainsi la possibilité d’afficher à l’écran 10 des questions les plus souvent demandées au programme :



En choisissant une des requêtes, l’utilisateur verra affiché à l’écran un brouillon des informations contenues dans la base de données selon la question :

